Fast NRZLM Encoding and Decoding Algorithm

Byte-oriented algorithms save time.

Goddard Space Flight Center, Greenbelt,
Maryland

A recently developed algorithm saves encoding and decoding time in the operation of data-communication systems that utilize the NRZM code, which is derived from the better-known non-return-to-zero-level (NRZL) code. This algorithm utilizes lookup tables that contain the results of routine encoding and decoding computations that would otherwise have to be performed repeatedly.

A stream of symbols in NRZM code is generated from an input stream of symbols in NRZL code. The NRZM code was originally developed as a means to convert a steady, high-level signal (a long sequence of 1111...111) into a variable signal (1010....10 or 0101.....01, depending on the choice of 0 or 1 for the initial state of the coding algorithm). The NRZM code provides signal-level transitions in an idle state when there is time for synchronization of encoding and decoding equipment.

An explanation of the nomenclature of algorithms for NRZM encoding and decoding is prerequisite to an explanation of the present innovative algorithm. In general, algorithms that transform streams of symbols between NRZL and NRZM codes are denoted collectively as "NRZLM" algorithms. Of these, subalgorithms that transform NRZL input streams into NRZM output streams are called "NRZM" algorithms, while subalgorithms that transform NRZM input streams into NRZL output streams are called "NRZL" algorithms.

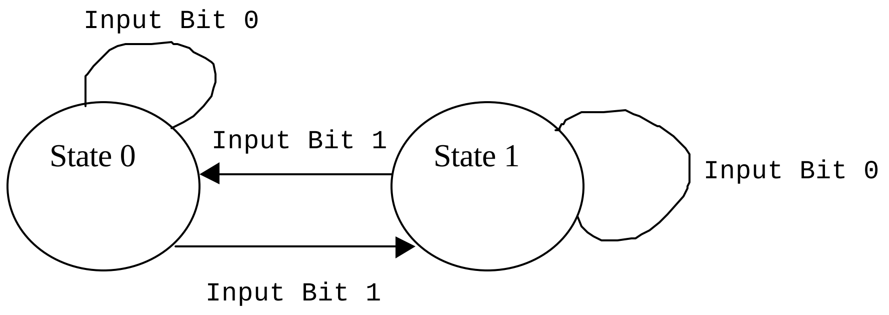
Before the present innovative NRZM algorithm was developed, transformations between NRZL and NRZM were effected by the Binary NRZLM algorithm, which is bit-oriented; that is, it operates on only one bit at a time. Even if an NRZL data source is byte-oriented, it is necessary to disassemble the NRZL bytes into bits, then encode the bits into NRZM one at a time by use of the Binary NRZM algorithm, then reassemble the NRZM-encoded bits into bytes. Similar considerations apply to use of the Binary NRZL algorithm to decode from NRZM back to NRZL.

The NRZM code and the Binary NRZM algorithm can be explained in terms of a finite-state automaton that can be in either of two states; 0 or 1 (see figure). These states correspond to output bits. If the automaton is in
either state and receives an input bit 0, it remains in that state. If the automaton is in either state and receives an input bit of 1, it changes to the other state. Thus, the output bit for a given input bit depends on the state of the decoder after receipt of the immediately preceding input bit; this state is called the "last state." The last state depends on the chosen initial state and on the sequence of input bits up through the immediately preceding bit.

Thus, in the Binary NRZM algorithm, it is necessary to go bit-by-bit through the entire sequence of preceding NRZL input bits to arrive at the output NRZM bit for a given input NRZL bit. Similarly, in the Binary NRZL algorithm, it is necessary to go bit-by-bit through the entire sequence of preceding NRZM input bits to arrive at the NRZL output bit for a given input NRZM bit.

The present innovative NRZLM algorithm is byte-oriented. It exploits the following observation: One can commence coding or decoding from any point in a sequence of input bits, without having to step through the entire sequence of preceding input bits, provided that one has some other way of knowing the last state immediately preceding that point. Thus, if bits in an input sequence are grouped into bytes, one can start to encode or decode at the beginning of any byte, provided that one knows the last state produced by the preceding byte or bytes.

In formulating this byte-oriented NRZLM algorithm, the results of coding and decoding operations are precomputed by the Binary NRZM and Binary NRZL algorithms and stored in lookup tables; these tables contain the output bytes and last states for all possible input bytes and preceding last states. Thus, instead of a long sequence of operations on individual bits, the encoding and decoding of each input byte involves only initialization by use of the last state from the preceding byte, followed by a table-lookup operation to find the output byte and another table-lookup operation to find the new last state.



A Two-State Automaton that makes transitions in response to an input bit of 1 implements the NRZM code and the Binary NRZM algorithm.

This work was done by Semion Kizhner and Timothy Ray of Goddard Space Flight Center. For further information, access the Technical Support Package (TSP) free on-line at www.nasatech.com under the Information Sciences category.

GSC-13825

NASA Tech Briefs
Engineering Solutions for Design & Manufacturing
December, 2000